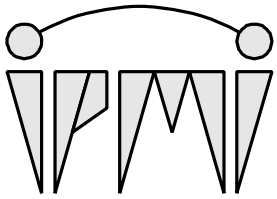

Errata & Update Summary

Intelligent Platform Management Interface Specification v0.9

Document Revision 3

August 09, 1998

Copyright © 1997, 1998 Intel Corporation
All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

INTEL DISCLAIMS ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. INTEL DOES NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

I^2C is a trademark of Phillips Semiconductors.
All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

I^2C is a two-wire communications bus/protocol developed by Philips. IPMB is a subset of the I^2C bus/protocol and was developed by Intel. Implementations of the I^2C bus/protocol or the IPMB bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel retains the right to make changes to this document at any time, without notice. Intel makes no warranty for the use of this document and assumes no responsibility for any error which may appear in the document nor does it make a commitment to update the information contained herein.

Introduction

This document presents a summary of the changes between the *Intelligent Platform Management Interface Specification* revision 0.16 and revision 0.17, and a listing of present errata on revision 0.17. Section, table, and figure references are given relative to the newer version of the specification, unless otherwise noted. Where examples are given, text additions are shown with double underlines, and text deletions are shown with strike-through.

As of this writing, revision 0.17 of the *Intelligent Platform Management Interface Specification* is available from the IPMI Web Site at:

http://developer.intel.com/design/servers/ipmi

Errata & Clarifications

Errata Revision 3, 8/9/98:

- **Section 20.1 KCS Interface Registers** : Errata. The base address for the KCS interface was incorrectly specified as CA0h instead of CA2h. Since it is possible that implementations may have already been produced using the CA0h address, the IPMI group has elected to adopt the CA0h address as a *secondary* address for the KCS interface.

  The statement "The default system I/O base address for the SMS Interface is CA0h" should be changed to read "The primary default system I/O base address for the SMS Interface is CA2h. CA0h can be used as a secondary address for the interface. System software should test for the KCS interface at the CA2h location before using the CA0h address."

Errata Revision 2, 6/17/98:

- **Section 5.1, Section 15.3** : **Clarifications**. The 'Event Generator' and 'Event Receiver' capabilities flags return in the Device Locator Record and the *Get Device ID* command refer to whether the controller is an 'IPMB Event Generator' or 'IPMB Event Receiver', respectively. That is, these bits reflect whether the device generates or receives events from the *IPMB.* Thus, a BMC that serves as an event receiver, but was only able to generates events internally (not as event messages on the IPMB) would be represented as an Event Receiver, but not as an Event Generator. Another way of looking at this is that the Event Generator bit indicates whether a controller accepts the *Set/Get Event Receiver* commands or not, whereas the Event Receiver bit indicates the controller can be the target for event messages.

- **Section 6.5.1** : Errata. The term 'Cold Reset' should be deleted from the statement. "A device that receives a 'Set Event Receiver' or Cold Reset command shall 're-arm' event generation for all its internal sensors."

- **Section 8.5 Get SEL Entry - Storage.43h** : Errata. The text states "the response returns the 14 bytes of data following the Record ID field in the SEL Event Record". The command actually returns all 16 bytes of the record. Thus, the specification should state "the response returns the 16 bytes of the SEL Event Record".

- **Table 9-1, SEL Event Record - Type 02h** : Errata. The descriptions on the Event Data bytes (bytes 14:16) include the phase "Per following table." - The correct text should be "See Table 7-5".

- **Table 10-2, Get SDR Repository Info - Storage.20h** : Description of Byte 15 bit 7 "SDR could not be written due to lack of space in the SEL" should be "SDR could not be written due to lack of space in the SDR Repository".

- **Section 12.11** : Clarifications. Request Byte 2 is allowed to be sent for analog or threshold based sensors, but will be ignored for those sensor types. A discrete sensor that only provides 'entire sensor' disable (instead of a per threshold disable) will return the 'entire sensor' setting regardless of the offset passed in the request.
- **Section 20.3.1** : Clarification. The "SMS_ATN" bit gets set when the BMC has an *unsolicited* message for System Management Software. For IPMI v0.9 the SMS_ATN bit indicates that there is a message available in the SMS message buffer.

Update Summary

Global Changes

- Changed occurrences of "IMB" to "IPMB"
- Changed occurrences of "Intelligent Management Bus" to "Intelligent Platform Management Bus"
- Changed copyright year to 1998, where necessary

Specific Changes

- **Section 3.2** - changed section title from "System Side Sensor Owner IDs" to "System Software IDs"
- **Table 5-1** - added note designation (**) to Cold Reset command and corresponding note to table: "This command is not required to return a response in all implementations." That is, an implementation of the Cold Reset command can elect to not return a response, yet still meet the IPMI specification.
- **Section 5.2** - added note on the purpose and platform-specific aspects of the cold reset command.
- **Section 6.3, paragraph 6** - Changed "Intelligent Management Bus v1.1 Communications Protocol" to "Intelligent Platform Management Bus v0.9 Communications Protocol"
- **Table 7-2** - added specification for which bits in request byte 1 hold the I^2C slave address.
- **Table 7-3** - added specification for which bits in response byte 1 hold the I^2C slave address.
- **Table 7-4** - changed field name "Event Type Code" to "Event Type" for terminology consistence.
- **Section 7.4, under 'Generator ID'** - Changed occurrence of 'IBM' to 'IPMB'.
- **Section 7.4, under 'Event Type'** - Added explanatory text "This is encoded using the Event/Reading Type Code."
- **Section 7.5, first paragraph** - Changed "Intelligent Management Bus v1.1 Communications Protocol" to "Intelligent Platform Management Bus v0.9 Communications Protocol"
- **Section 7.7, first paragraph** - Changed as follows:

  The contents of the Event Data field in an Event Request Message (Event Message) is dependent on the 'event trigger class' for the event generated by the sensor. The trigger class specifies whether the sensor event 'trigger' is threshold based, digital, discrete, or OEM defined. Each Event Type is associated with a trigger class. Thus, an application can determine the trigger class, and hence the corresponding Event Data format, from the Event/Reading Type Code that was received in the Event Type field in the Event Message. See section 17.1, *Event/Reading Type Codes*, for more information.

- **Table 7-5** - Changed occurrences of “Event Trigger” to “Event/Reading Type Code” for terminology consistence.
- **Table 8-2** - Changed response byte 15 definition to as follows:

•	• Operation Support • Overflow Flag. 1=Events have been dropped due to lack of space in the SEL. • reserved. Write as 000 • 1=Delete SEL command supported • 1=Partial Add SEL Entry command supported • 1=Reserve SEL command supported • 1=Get SEL Allocation Information command supported

- **Table 9-1, byte 13** - Changed field name “Event Trigger” to “Event Type” for terminology consistence.
- **Table 9-1, byte 13** - Added explanatory text to description. “The Event Type field is encoded using the Event/Reading Type Code. See section 17.1, *Event/Reading Type Codes.*”
- **Table 10-2** - Cut and paste error. Row referred to ‘SEL’ when it should have referred to ‘SDR’. Changed response byte 15 definition as follows:

15	Operation Support	
	7	Overflow Flag. 1=SDR could not be written due to lack of space in the SEL.
	6:4	reserved. Write as 000
	3	1=Delete SDR command supported
	2	1=Partial Add SDR command supported
	1	1=Reserve SDR command supported
	0	1=Get SDR Allocation Information command supported

- **Section 12-1** - Deleted word ‘devices’ from first sentence as follows:
- Static Sensor Devices are defined as sensors that have their Sensor Data Records added to the SDR Repository when the device is configured into the system.
- **Table 12-5** - Added “(FFh = reserved)” to request byte 1 definition.
- **Section 12.6** - Added explanatory text to first paragraph:

  The positive hysterisis value is used for positive-going thresholds, while the negative hysterisis value is used for negative-going thresholds. The positive hysterisis value is *subtracted* from the threshold to set the point at which the event clears, while the negative hysterisis value is added to the threshold. The event will clear when the value equals the ‘hysterisis threshold’.

- **Table 12-6, 12-7, 12-8, 12-9, 12-10, 12-11, 12-12, 12-14** - Added “(FFh = reserved)” to request byte 1 definition.
- **Table 12-14** - Changed occurrences of “at or below” to “below”, and “at or above” to “above” for consistency with text. Text states that events are generated when a threshold is *exceeded* implying > comparisons (and correspondingly, < comparisons) rather than □ , □ comparisons. Also changed text indicating which type of response is used according to sensor event generation type (threshold based, digital, discrete, or OEM) for bytes 2:(3) - as follows:

2:(3)	For sensors with threshold based events: Byte 1: 7 reserved. Returned as '1'. 6 0 = Sensor disabled (not scanning) 5 1 = below lower non-recoverable 4 1 = above upper non-recoverable 3 1= below lw (lower non-critical) 2 1= above uw (upper non-critical) 1 1= below lf (lower critical) 0 1= above uf (upper critical) For sensors with digital events: Byte 1: 7 reserved. Returned as '1'. 6 0 = Sensor disabled (not scanning) 5:1 reserved. Returned as '0'. 0 1 = event condition active (event triggered) 0 = event condition inactive For sensors with discrete events: Byte 1: Event mask, LS byte. Bits indicate offsets to discrete event, starting with bit 0 of Byte 1 through bit 6 of Byte 2. 7:0 Offsets 7 to 0. 1 = event condition active (event triggered) 0 = event condition inactive Byte 2: 7 0 = Sensor disabled (not scanning) 6:0 Offsets 14 to 8. 1 = event condition active (event triggered) 0 = event condition inactive

- **Table 12-15** - Added "(FFh = reserved)" to request byte 1 definition.
- **Table 12-15** - Changed occurrences of "at or below" to "below", and "at or above" to "above" for consistency with text. Text states that events are generated when a threshold is *exceeded* implying > comparisons (and correspondingly, < comparisons) rather than □ , □ comparisons. Also changed text indicating which type of response is used according to sensor event generation type (threshold based, digital, discrete, or OEM), and text to help show which bit positions correspond to which Event/Reading offsets for bytes 2:(3) - as follows:

2:(3)	Sensor reading For threshold-based or 'analog' reading sensors Byte 1: byte of reading. Byte 2: Present threshold comparison status 7 reserved. Returned as '1'. 6 0 = Sensor disabled (not scanning) 5 1 = below lower non-recoverable 4 1 = above upper non-recoverable 3 1= below lw (lower non-critical) 2 1= above uw (upper non-critical) 1 1= below lf (lower critical) 0 1= above uf (upper critical) For digital reading sensors Byte 1: 7 reserved. Returned as '1'. 6 0 = sensor disabled (not scanning). 5:1 reserved. Returned as 0 0 offset for digital status. Relative to Event/Reading type code. For discrete reading sensors Byte 1: Event mask, LS byte. Bits indicate offsets to discrete event, relative to the Event/Reading type code for the sensor. Starting with bit 0 of Byte 1 through bit 6 of Byte 2. 7:0 Bit field corresponding to Event/Reading Offsets 7 to 0, respectively. 1 = event condition active (event triggered) 0 = event condition inactive Byte 2: (Mandatory for discrete sensors) 7 0 = Sensor disabled (not scanning) 6:0 Bit field corresponding to Event/Reading Offsets 14 to 8, respectively. 1 = event condition active (event triggered) 0 = event condition inactive

- **Table 15-1, byte 13** - Added text explaining initialization of sensor hysterisis to bit 7 specification, as follows:

  7 1 = sensor hysterisis is settable (The Threshold Init Required bit must also be 1 for the initialization agent to set hysterisis. A device that requires settable hysterisis, but does not require threshold initialization, should set the settable threshold mask (below) to 0.

  A device that has settable hysterisis, but does not require hysterisis initialization, should define a 'no-op' hysterisis value (FFh recommended) and have that value used in the hysterisis values in this SDR. The device would accept, but not change, the threshold value upon receiving that hysterisis value from the initialization agent.)

- **Table 15-1, byte 13** - Changed "Event Trigger" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-1, byte 15** - Changed "Event / Reading Base Type" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-1, byte 16** - Changed "Event Trigger Base" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-1, byte 18** - Changed field name from "Event Reading Masks" to "Reading Mask" for terminology consistency. Also changed "Event Trigger Base" to "Event/Reading Type Code".

- **Table 15-1, byte 48** - Changed field name from "Type/Length Code" to "ID String Type/Length Code" to better denote that the string is intended more as an additional identifier, rather than a descriptive name. Also change 'Name' to 'ID' in the description.

- **Table 15-1, byte 49** - Changed "String Bytes" to "ID String Bytes". Also changed "Name" to "ID String" in description.

- **Table 15-1, byte 49** - Added note: "Note: the SDR can be implemented as a fixed length record. Bytes beyond the ID string bytes are unspecified and should be ignored." - This means that the Record Length field (byte 5) can define a record size that extends beyond the last byte in the ID string.

- **Table 15-1, note 2** - Corrected example to show correct definition of slave address bits, as follows:

    2. 7-bit I^2C Slave Address field. By convention, we normally designate an I^2C slave address as an eight-bit number with the least-significant bit always 0. E.g. 20h = 00100000b. The 7-bit Slave Address field holds the most-significant 7 bits of this value. E.g. 0010000b.

- **Table 15-2, byte 13** - Changed "Event Trigger Base" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-2, byte 14** - Changed occurrences of "Instance Name" to "ID String Instance" for terminology consistency, and changed string in example from "Temperature" to "Temp" to better convey that this string is not expected to be a full name - but just an additional identifier. Changes were made as follows:

14 15	Sensor Record Sharing	2	Byte 1: ID String Instance Modifier Type 5:4: 0 = numeric 1 = alpha Share Count 3:0 Share count (number of sensors sharing this record). Sensor numbers are contiguous starting with the sensor number specified by the *Sensor Number* field for this record. Byte 2: Entity ID Sharing 7 0 = Entity ID same for all shared records 1 = Entity ID increments for each shared record 6:0 ID String Instance Modifier Offset Multiple Digital/Discrete sensors can share the same sensor data record. The ID String Instance Modifier and Modifier Offset are used to modify the Sensor ID String as follows: Suppose: sensor ID is "Temp " for 'Temperature Sensor', share count = 3, ID string instance modifier = numeric, instance modifier offset = 5 - then the sensors could be identified as: Temp 5, Temp 6, Temp 7 If the modifier = alpha, and offset = 26, then the sensors could be identified as: Temp AA, Temp AB, Temp AC (alpha characters are considered to be base 26 for ASCII)

- **Table 15-2, byte 17** - Changed "Event Trigger Base" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-2, byte 18** - Changed "Event Trigger Base" to "Event/Reading Type Code" for terminology consistency.

- **Table 15-2, byte 27** - Changed field name from "Type/Length Code" to "ID String Type/Length Code" to better denote that the string is intended more as an additional identifier, rather than a descriptive name. Also change 'Name' to 'ID' in the description.

- **Table 15-2, byte 28** - Changed "String Bytes" to "ID String Bytes". Also changed "Name" to "ID String" in description.

- **Table 15-2, note 1** - Corrected example to show correct definition of slave address bits, as follows:

    1. 7-bit I^2C Slave Address field. By convention, we normally designate an I^2C slave address as an eight-bit number with the least-significant bit always 0. E.g. 20h = 00100000b. The 7-bit Slave Address field holds the most-significant 7 bits of this value. E.g. 0010000b.

- **Table 15-3** - Typo. Changed column title "Enity ID" to "Entity ID".
- **Table 15-4, byte** 7 - Added text explaining bits 5:4 usage when device type is a FRU Device, as follows:

  5:4 Private bus ID if bus = Private. LUN if device is FRU Device accessed via Read/Write FRU Inventory commands at non-zero LUN. 00b otherwise.

- **Table 15-4, byte 16** - Changed field name from "Type/Length Code" to "ID String Type/Length Code" to better denote that the string is intended more as an additional identifier, rather than a descriptive name. Also change 'Name' to 'ID' in the description.
- **Table 15-4, byte 17** - Changed "String Bytes" to "ID String Bytes". Also changed "Name" to "ID String" in description.
- **Table 15-4, note 1** - Corrected example to show correct definition of slave address bits, as follows:

  1. 7-bit I^2C Slave Address field. By convention, we normally designate an I^2C slave address as an eight-bit number with the least-significant bit always 0. E.g. 20h = 00100000b. The 7-bit Slave Address field holds the most-significant 7 bits of this value. E.g. 0010000b.

- **Section 16.2, under "Sensor Specific".** Added text explaining that the E6h Event/Reading Type code is only used in Event Messages. Also changed term "enumeration" to "offsets" for terminology consistency.
- **Section 17.1** - Added explanatory text and made changes for terminology consistency. See Appendix A.
- **Table 17-3** - Changed column 2 title from "Sensor Event Type" to "Sensor Type Code" for terminology consistency. Also changed column 3 title from "Offset" to "Sensor-specific Offset" to better indicate that those offset values are used when the Event/Reading Type code is "sensor-specific" (E6h or E7h).
- **Table 17-3, under Processor (Sensor Type = 07h)** - Changed "Presence detected" to "Processor Presence detected". Added new offset "Terminator Presence Detected" - as follows:

Processor	07h	00h	0Bh	IERR
		01h	0Bh	Thermal Trip
		02h	0Bh	FRB1/BIST failure
		03h	0Bh	FRB2/Hang in POST failure
		04h	0Bh	FRB3/Processor Startup/Initialization failure (CPU didn't start)
			0Bh	Configuration Error (for DMI)
		05h	0Bh	DM BIOS 'Uncorrectable CPU-complex Error'
		06h	15h	Processor Presence detected
		07h	0Bh	Processor disabled
		08h	0Ch	Terminator Presence Detected
		09h	0Bh	

- **Table 17-3, under Power Unit** - Added two new offsets, as follows:

Power Unit	09h	00h	83h	Power Off / Power Down
		01h	97h	Power Cycle
		02h	98h	240VA Power Down
		03h	99h	Interlock Power Down
		04h	9Ah	A/C lost
		05h	9Dh	Soft Power Control Failure (unit did not respond to request to turn on)
				Failure detected
		06h	9Eh	

- **Table 17-3, 3rd row from end** - Added new 'Terminator' sensor type:

Terminator	1Ch	-	9Ch	-

- **Section 18.2, paragraph 1** - Changed "Write I^2C, Read I^2C, and Write-Read I^2C" to "Master Write I^2C, Master Read I^2C, and Master Write-Read I^2C" to reflect actual command names.
- **Section 18.2, paragraph 4** - Corrected "up to three private busses" to "up to four private busses".
- **Section 18.6 - NEW SECTION**. Added text explaining how system software can check for message buffer support. As follows:

18.6 Testing for Optional Message Buffer Support

System software must test for SMM or Event Message buffer support. If software issues a *Get BMC Global Enables* command, and finds the desired buffer enabled, it can assume the controller supports that message buffer. Otherwise, it must test by attempting to enable the desired message buffer.

If the BMC does not support the desired buffer, it shall return an illegal parameter error completion code when an attempt is made to enable the respective buffer using the *Set BMC Global Enables* command. An error completion code shall also be returned if an attempt is made to enable Event Message Buffer SMIs when that option is not supported.

- **Table 18-2, Commands 1Ch, 1Dh, 30h, and 31h -** Added text explaining error conditions and bit states when optional message buffers are not implemented. See Appendix B.

APPENDIX A
Section 17.1 Changes

17.1 Event/Reading Type Codes

Event/Reading Type codes are used in SDRs and Event Messages to indicate the trigger type for an event. These codes are also used in SDRs to indicate what types of *present reading* the sensor provides.

Event/Reading Type Codes are used to specify a particular enumeration that identifies a set of possible events that can be generated by a sensor. For digital and discrete sensors, the specification of an Event/Reading Type code enumeration also indicates the type of reading the sensor provides.

Sensors fall into the following classes:

Sensor Classes:

Digital Only two states possible. *Digital* sensors can have only two possible states, reflected using the offset values 0 or 1. These values are returned as the sensor reading via a *Get Sensor Reading* command, are returned in event messages, and are used for event configuration. To get the full meaning of the reading or event, the offset must be combined with the corresponding Event/Reading Type Code for the sensor.

For example, from Table 17-2, 04h is the Event/Reading code for a digital sensor that has the possible generic states "Predictive Failure deasserted" and "Predictive Failure asserted". If a *Get Sensor Reading* command returns an offset of '0', then that means the present state is "Predictive Failure deasserted".

Discrete Multiple states possible. *Discrete* sensors can contain more than two possible states (up to 15). For discrete sensors, the *Get Sensor Reading* command returns a bit field where each bit reflects a different state. It is possible for a discrete sensor to have more than one state active at a time.

Discrete sensors can be designed to provide either *Generic* or *Sensor-specific* states. The Event/Reading Type Codes in Table 17-2 are used to specify the particular set of possible *Generic* states for a discrete sensor. Generic states may be applicable to many types of sensors - that is, a temperature sensor and a voltage sensor may both be implemented that return Severity states (Event/Reading Type Code 07h).

For example, Event/Reading Type Code 0Bh indicates a discrete sensor that could return one of three possible states "Redundancy Regained, Redundancy Lost, or Redundancy Degraded". The offsets from the table correspond to the bit positions in the *Get Sensor Reading* command. The offset values are also used in event messages and in event configuration. (Note, Event Messages reflect only one event at a time. Thus, Event Messages do not return a bit field, just a single offset value corresponding to a single event.)

Sensor-specific states, however, are tied to a particular sensor type. The sensor-specific offsets are specified in the sensor types table: *Table 17-3, Sensor Type Codes*. When the sensor-specific offsets are used in the SDR for a sensor, the Event/Reading Type Code E7h is used. This code is also used in Event Messages triggered by the *assertion* of a sensor-specific state. Event/Reading Type Code E6h is used only in Event Messages

when a event is triggered by the *deassertion* of a sensor-specific state.

Threshold 'Threshold based'. Triggers on reading comparison to threshold values. *Threshold* enumerations may be considered a special case of the discrete sensor type. The Event/Reading Type Code for threshold-based sensors is specified in *Table 17-2, Event/Reading Type Codes*, below. The offsets specify each particular possible threshold state.

Threshold-based sensors return a different response to the *Get Sensor Reading* command than discrete sensors. The offsets The *Get Sensor Reading* command for a threshold-based sensor contains the present 'analog' reading from the sensor along in addition to the discrete threshold comparison status bit field.

OEM Special case of discrete where the meaning of the states (offsets) is OEM defined.

Table 17-1, Event/Reading Type Code Ranges

Event/Reading Type Code category	Event/Reading Type Code Range	Sensor Class	Description
unspecified	00h	n/a	Event/Reading Type unspecified.
Generic	01h-0Bh	digital, discrete, or threshold-based	Range for Generic digital, discrete, or threshold Event/Reading Type codes, as specified in Table 17-2, Event/Reading Type Codes, below.
Sensor Specific	E6h	discrete only	Standard Sensor specific enumeration - only used in Event Messages to report sensor-specific deassertion events.
Sensor Specific	E7h	discrete only	Standard Sensor specific enumeration - used to report sensor-specific assertion events, and to provide sensor-specific discrete readings and event status.
OEM	E8h-Efh	OEM	OEM defined enumeration

Codes that are not specified in this table, or the following tables in this section, are *reserved*.

Table 17-2, Event/Reading Type Codes

Generic Event/Reading Type Code	Event/Reading Class	Generic Offset	Description
			THRESHOLD BASED STATES
01h	Threshold	00h	Lower Non-critical - going low
		01h	Lower Non-critical - going high
		02h	Lower Critical - going low
		03h	Lower Critical - going high
		04h	Lower Non-recoverable - going low
		05h	Lower Non-recoverable - going high
		06h	Upper Non-critical - going low
		07h	Upper Non-critical - going high
		08h	Upper Critical - going low
		09h	Upper Critical - going high
		0Ah	Upper Non-recoverable - going low
		0Bh	Upper Non-recoverable - going high
			DMI-based "Usage State" STATES
02h	Discrete	00h	Transition to Idle
		01h	Transition to Active
		02h	Transition to Busy

Generic Event/Reading Type Code	Event/Reading Class	Generic Offset	Description
	03h		DIGITAL/DISCRETE EVENT STATES
03h	Digital	00h	State Deasserted
		01h	State Asserted
04h	Digital	00h	Predictive Failure deasserted
		01h	Predictive Failure asserted
05h	Digital	00h	Limit Not Exceeded
		01h	Limit Exceeded
06h	Digital	00h	Performance Met
		01h	Performance Lags
			DMI-based SEVERITY EVENT STATES
07h	Discrete	00h	transition to OK
		01h	transition to Non-Critical from OK
		02h	transition to Critical from less severe
		03h	transition to Non-recoverable from less severe
		04h	transition to Non-Critical from more severe
		05h	transition to Critical from Non-recoverable
		06h	transition to Non-recoverable
		07h	Monitor
		08h	Informational
			DMI-based AVAILABILITY STATUS STATES
08h	Digital	00h	Device Removed
		01h	Device Inserted
09h	Digital	00h	Device Disabled
		01h	Device Enabled
0Ah	Discrete	00h	transition to Running
		01h	transition to In Test
		02h	transition to Power Off
		03h	transition to On Line
		04h	transition to Off Line
		05h	transition to Off Duty
		06h	transition to Degraded
		07h	transition to Power Save
		08h	Install Error
			Other AVAILABILITY STATUS STATES
0Bh	Discrete	00h	Redundancy Regained
		01h	Redundancy Lost
		02h	Redundancy Degraded

APPENDIX B
Table 18-2 Changes

Code	Command	Net Fn	LUN	Request (Command) / Response Data	POR default
1Ch	Set BMC Global Enables	App	00	Request: byte 1: 7 - 0= disable all BMC SMIs 6 - reserved. Write as '0'. 5 - 1 = enable SMM Message Buffer 0 = disable SMM Message Buffer. Error completion code returned if written as '1' and SMM Message Buffer not supported. 4 - 1 = Enable Event Message Buffer (Route Event Requests to Event Message Buffer) 0 = disable Event Message Buffer. Error completion code returned if written as '1' and SMM Message Buffer not supported. 3 - reserved. Write as '0' 1 0 = enable event logging. 1 = disable event logging. 0 - 1 = enable Event Message Buffer SMIs. Error completion code returned if written as '1' and Event Message Buffer SMIs not supported. 0 = disable Event Message Buffer SMIs.	00h
				Response: byte 1 - Completion Code	
1Dh	Get BMC Global Enables	App	00	Request: -	
				Response: byte 1 - Completion Code byte 2: 7 - 0= all BMC SMIs disabled 6 - reserved. Ignore on read. 5 - 1 = SMM Message Buffer enabled 0 = SMM Message Buffer disabled. Also returns 0 if SMM Message buffer not implemented. 4 - 1 = Event Requests routed to Event Message Buffer. Returns 0 if Event Message buffer not implemented. 0 = Event Message Buffer disabled. Also returns 0 if SMM Message buffer not implemented. 3:2 - reserved. Ignore on read 1 - 0 = event logging enabled. 1 = event logging disabled. 0 - 1 = Event Message Buffer SMIs enabled. Reserved, ignore on read, (return 0) if Event Message buffer not implemented.	
30h	Clear Interrupt Flags	App	00	Request: byte 1 - flags (0 = clear flag, 1=no action) 7:6 - reserved. Write as '1'. 5 - SMM Buffer Full (reserved, write as '1', if SMM message buffer not implemented) 4:3 - reserved. Write as '1'. 2 - SMS Message Buffer Full 1 - reserved. Write as '1'. 0 - Event Message Buffer Full	
				Response: byte 1 - Completion Code	

Last Page